IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Kevin Curtis Griffin et al.	Art Unit:	2186
Application No.:	10/758,484	Examiner:	Jonathan A. Barton
Filed:	January 15, 2004	Atty. Docket No.:	ROC920030367US1
For:	ASYNCHRONOUS HYBRID MIRRORING SYSTEM		

DECLARATION OF KEVIN CURTIS GRIFFIN UNDER 37 CFR § 1.131

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

I, Kevin Curtis Griffin, hereby declare and state:

1. I am an inventor of the above-identified U.S. Patent Application.

2. Prior to October 28, 2003, my co-inventors and I, Scott Dennis Helt, Michael James McDermott, Glen W. Nelson and Mark Philip Piazza, conceived of a concept of efficiently and reliably mirroring data of a primary system to a backup system. In one respect, our concept provides a hybrid approach that enjoys benefits of both the fast processing of asynchronous mirroring and the data integrity of synchronous mirroring. More particularly, a number of update requests are organized into groups at both the primary and backup systems. The respective groups are processed synchronously to preserve sequential ordering. The updates of each group, however, are processed concurrently, or substantially at the same time and without regard to order. This allows improved processing times with regard to the requests of each group.

3. Put another way, the requests of a first group are processed concurrently before a subsequent request issues. This feature maintains data integrity and sequential ordering. As such, all update requests in one group complete before a request of a subsequent group issues and completes. Concurrent issuance of the requests in each group allows relatively quick processing. The sequential, ordered processing of the respective groups provides relatively good data integrity.

4. In one implementation of our concept, an operating system executing program code present on at least one of the primary and backup systems categorizes incoming requests according to a group number. Update requests assigned the same group number will issue concurrently after all requests of a preceding group have completed. As such, subsequently created group numbers may be incremented automatically to maintain chronology and desired, sequential ordering.

5. Also, with our concept, memory accessible to the operating system of either or both the primary and backup systems may track a status that is indicative of whether a group is currently active or in progress. An active group status may cause a newly received request to be properly associated with the current group for processing considerations. Conversely, an inactive status may prompt the creation of a new group. The memory may also maintain and allow updating of a count indicative of how many outstanding requests remain in a group. Such a count is helpful in regulating the creation and ordered execution of respective groups.

6. Prior to October 28, 2003, my co-inventors and I submitted an invention disclosure form to our employer, International Business Machines Corporation, which described our improved apparatus, program products and methods. This form was created and last modified prior to October 28, 2003. A copy of this form (with portions thereof redacted) is attached hereto as Exhibit A.

7. From the period prior to October 28, 2003, until the filing of the patent application on January 15, 2004, my co-inventors and I were diligent in constructively reducing our invention to practice. During this time period, we met with inside counsel at International Business Machines Corporation to explain our invention, participated in telephone calls with the outside counsel assigned to prepare the patent application, provided supplemental materials to outside counsel to assist in preparing the patent application, reviewed a draft of the patent application prepared by outside counsel, supplied comments and suggested revisions to outside counsel, and reviewed and executed a final draft of the patent application.

8. The statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true; further, these statements are made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001, Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the above-referenced application or any patent issuing thereon.

Respectfully submitted,

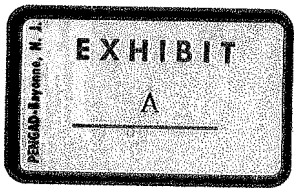
4/19/06
Date

Kevin Curtis Griffin

Disclosure

Prepared for and/or by an IBM Attorney - IBM Confidential

Created By Mike McDermott
Last Modified By Mike McDermott



Required fields are marked with the asterisk (*) and must be filled in to complete the form .

***Title of disclosure (in English)**

Asynchronous mirroring for high performance while preserving order dependent updates for data integrity

Summary

Status		Submitted
Final Deadline		
Final Deadline Reason		
*Processing Location		Rochester
*Functional Area	select	(2R2) 2R2 - SG - eServer Hardware Enablement - Henry Hocraffer
Attorney/Patent Professional		James R Nock/Rochester/IBM
IDT Team	select	Rich Diedrich/Rochester/IBM Eric Barsness/Rochester/IBM Michael Branson/Rochester/IBM Bill Berg/Rochester/IBM James Carey/Rochester/IBM Scott Gerard/Rochester/IBM Greg Leibfried/Rochester/IBM Gary Ricard/Rochester/IBM Bill Schmidt/Rochester/IBM Blair Wyman/Rochester/IBM James R Nock/Rochester/IBM
Submitted Date		
*Owning Division	select	SG
Incentive Program		
Lab		
*Technology Code		231
PVT Score		

Inventors with a Blue Pages entry

Inventors: Mike McDermott/Rochester/IBM, Scott Helt/Rochester/IBM, Kevin Griffin/Rochester/IBM, Glen Nelson/Rochester/IBM, Mark Piazza/Rochester/IBM

Inventor Name	Inventor Serial	Div/Dept	Inventor Phone	Manager Name
> McDermott, Michael J.	562684	7T/47WA	553	
Helt, Scott D.	542472	7T/47WA	553	
Griffin, Kevin C. (Kevin C)	288896	7T/47WA	553	
Nelson, Glen W.	767629	7T/47WA	553	
Piazza, Mark P.	778952	7T/47WA	553	

> denotes primary contact

Inventors without a Blue Pages entry

IDT Selection

Attorney/Patent Professional	James R Nock/Rochester/IBM
IDT Team	Rich Diedrich/Rochester/IBM Eric Barsness/Rochester/IBM Michael Branson/Rochester/IBM Bill Berg/Rochester/IBM James Carey/Rochester/IBM Scott Gerard/Rochester/IBM Greg Leibfried/Rochester/IBM Gary Ricard/Rochester/IBM Bill Schmidt/Rochester/IBM Blair Wyman/Rochester/IBM James R Nock/Rochester/IBM
Response Due to IP&L	

***Main Idea**

1. Background: What is the problem solved by your invention? Describe known solutions to this problem (if any). What are the drawbacks of such known solutions, or why is an additional solution required? Cite any relevant technical documents or references.

The invention applies to situation of mirroring data from a source system to a target system located at a distance to provide both availability and site disaster recovery. When applications make updates to the data, the updates must be made on both the source copy and on the target copy. Given that updates sent to the target copy are significanly slower, the problems to solve are:
o Maintain order of updates to the target copy so the target copy is usable (has integrity) at any point in time so that the target copy can become the source copy if the source copy fails.
o Achieve adequate performance to avoid impacting application performance.

The mirroring of updates to a target system is transparent to the application issuing updates. The mirroring function sending the updates to the target and performing the updates on the target, does not know which updates are order dependent.

Therefore, without knowledge of order dependency among updates, the alternatives are:
o Perform updates synchronously to ensure order of updates is maintained which keeps the the target copy usable, but degrades performance. The application on the source receives update completion when the update completes on both source and target.

o Perform updates asynchronously and partially in parallel on target copy which provides better performance, but the target copy is unusable until quiesced. The application on the source receives update completion when the update completes on the source and the update is received on the target.

2. Summary of Invention: Briefly describe the core idea of your invention (saving the details for questions #3 below). Describe the advantage(s) of using your invention instead of the known solutions described above.

The concept is that at an instance in time, all of outstanding updates in progress on source system have no order dependency. When there is order dependency, the application must wait for completion of its previous update before issuing its next update. This invention conceptually takes a snapshot of

outstanding updates at a point in time, then allows the target to perform these updates in any order.

The advantages are:
o Updates are performed asynchronously and partially in parallel providing better performance.
o The target copy is always usable and therefore can become the source copy at any point in time without requiring a successful quiesce.

3. Description: Describe how your invention works, and how it could be implemented, using text, diagrams and flow charts as appropriate.

The invention assigns outstanding updates to groups such that the target can perform all updates in a group in any order,
but cannot perform updates from the next group until all updates from the previous group are complete.
Each group is an extended snapshot which contains a set of updates without any order dependency.
Therefore, the target can perform all updates in a group totally in parallel in any order.

The steps on the source system are:
1) When initialize:
    - Set no group in progress
    - Set current group number to 0
2) When an update is issued:
    - If no group in progress:
        - Set group in progress
        - Increment current group number
    - Assign current group number to this update request
    - Send update request to target
    - Issue update request on source
    - When update request completes on source, return to client
3) When update request completes on source and update received on target:
    - If first update to complete for current group
      (group in progress AND same group number)
      (Note: Tuning possibility: defer returning completion to application
      to accumulate more updates in current group.)
        - Set no group in progress. Causes next update to start a new group
    - If subsequent update to complete
      (no group in progress OR different group number than current group)
        - Nothing extra to do
    - Return update completion to application.
      (Note: Since processing asynchronously, do NOT wait for update to complete on target.
      If update fails on target, target is suspended.)

The target system processing is dependent that the update requests are processed in the same order sent by source system to ensure that the target receives all updates in one group before receiving any updates from next group. The transport mechanism uses its own sequence number to ensure this order. The alternative disclosed in ROC8-2003-0181 has this same dependency.

The steps on the target system are:
1) When initialize:
    - Set no group in progress
    - Set count of updates in group to 0
2) When receive an update request from source:
    - If no group in progress

- Set group in progress
- Set current group number to group number from update request
- Increment count of update requests for current group
- Issue the update request

- If group in progress
  - If group number for update request same as current group number:
    - Increment count of update requests for current group
    - Issue the update request
  - If group number for update request is greater than current group number:
    - If current group count of update requests is 0
      - Start new group; set group in progress
      - Set current group number to group number from update request
      - Set count of updates in group to 1
    - If current group count of updates is greater than zero.
      - Hold this update request
  - If group number for update request is less than current group number:
    - Cannot occur; must be defect in implementation

3) When an update request completes:
- Decrement count of update requests for current group
- If count of update requests is 0 AND held update requests exist:
  - Set no group in progress (redundant because count of 0 will start new group)
  - Release the held update requests; re-process

On the source, the checking of group in progress, setting group in progress, setting group not in prgress, and setting the current group number are all serialized.
Similarly, on the target, the checking of group in progress, setting group in progress, setting group not in prgress, incremeting the current group count and decrementing the current group count are all serialized.